<u>IN THE CLAIMS</u>:

Please amend the claims as follows:

1. (Currently Amended) A method for ~~parsing documents in~~ query processing <u>by using a streaming application programming interface (API) for a mark-up language data stream of a textual document</u>, said method comprising:

producing ~~at least one~~<u>, by said streaming API for a mark-up language data stream, an ordered</u> index of ~~a document written in a mark-up language, wherein said mark-up language comprises any of HTML and XML~~ <u>all textual elements corresponding to their order in said mark-up language data stream, said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements</u>;

~~corresponding said index to said document;~~

scanning ~~said document using a~~<u>, by a</u> processor<u>, all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers</u>;

~~using a parser that is external to said index to selectively skip portions of said document based on instructions from said index, wherein the skipped portions of said document comprise portions irrelevant to said query, and wherein said index comprises a plurality of elements representing textual categories of said query; and~~

~~saving said textual categories into a buffer, wherein said buffer is external to said index, said parser, and said processor,~~

~~wherein said instructions match said elements to said query and if said elements do not match said query, then said parser uses said index to skip the portions of the document corresponding to unmatched elements,~~

~~wherein said each of said elements corresponds to a position in said document,~~

~~wherein said position comprises an end position,~~

~~wherein said index uses said end position as a marker for determining where to resume scanning said document upon skipping said portions of said document,~~

~~wherein said elements comprise sub-elements representing textual sub-categories of said~~

~~query, and~~

~~wherein said sub-elements updates said position in said document upon skipping said portions of said document and resuming scanning of said document~~

parsing a matched textual element, if a tag identifier, corresponding to said matched textual element, matches said query; and

skipping an unmatched textual element for parsing, if a tag identifier, corresponding to said unmatched textual element, does not match said query.

2-12. (Canceled).

13. (Currently Amended) A system for ~~parsing documents in~~ query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document, said system comprising:

~~at least one~~ an ordered index ~~corresponding to a document written in a mark-up language, wherein said mark-up language comprise any of HTML and XML~~ of all textual elements corresponding to their order in said mark-up language data stream, said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements and being produced by said streaming API for a mark-up language data stream;

a processor ~~operable for scanning~~ ~~said document~~ adapted to scan all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers; and

~~a parser that is external to said index and operable for selectively skipping portions of said document based on instructions from said index; and~~

~~a buffer operable for saving said textual categories,~~

~~wherein the skipped portions of said document comprise portions irrelevant to said query,~~

~~wherein said index comprises a plurality of elements representing textual categories of said query,~~

~~wherein said instructions match said elements to said query and if said elements do not match said query, then said parser uses said index to skip the portions of the document~~

~~corresponding to unmatched elements,~~

~~wherein said each of said elements corresponds to a position in said document,~~

~~wherein said position comprises an end position,~~

~~wherein said index uses said end position as a marker for determining where to resume scanning said document upon skipping said portions of said document,~~

~~wherein said elements comprise sub-elements representing textual sub-categories of said query, and~~

~~wherein said sub-elements updates said position in said document upon skipping said portions of said document and resuming scanning of said document~~

a parser adapted to parse a matched textual element, if a tag identifier corresponding to said matched textual element, matches said query, and to skip an unmatched textual element for parsing, if a tag identifier, corresponding to said unmatched textual element, does not match said query.

14-24. (Canceled).

25. (Currently Amended) A program storage device readable by computer comprising a program of instructions executable by said computer to perform a method for ~~parsing documents in~~ query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document, said method comprising:

producing ~~at least one~~, in said streaming API for a mark-up language data stream, an ordered index of ~~a document written in a mark-up language, wherein said mark-up language comprises any of HTML and XML~~ all textual elements corresponding to their order in said mark-up language data stream, said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements;

~~corresponding said index to said document;~~

scanning ~~said document using a~~, by a processor, all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers;

~~using a parser that is external to said index to selectively skip portions of said document~~

~~based on instructions from said index, wherein the skipped portions of said document comprise portions irrelevant to said query, and wherein said index comprises a plurality of elements representing textual categories of said query; and~~

~~saving said textual categories into a buffer, wherein said buffer is external to said index, said parser, and said processor,~~

~~wherein said instructions match said elements to said query and if said elements do not match said query, then said parser uses said index to skip the portions of the document corresponding to unmatched elements,~~

~~wherein said each of said elements corresponds to a position in said document,~~

~~wherein said position comprises an end position,~~

~~wherein said index uses said end position as a marker for determining where to resume scanning said document upon skipping said portions of said document,~~

~~wherein said elements comprise sub-elements representing textual sub-categories of said query, and~~

~~wherein said sub-elements updates said position in said document upon skipping said portions of said document and resuming scanning of said document~~

parsing a matched textual element, if a tag identifier, corresponding to said matched textual element, matches said query; and

skipping an unmatched textual element for parsing, if a tag identifier, corresponding to said unmatched textual element, does not match said query.

26-36. (Canceled).

37. (Currently Amended) A system for ~~efficiently parsing documents in~~ query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document, said system ~~consisting essentially of~~ comprising:

an ordered index ~~corresponding to a document written in a mark-up language, wherein said index comprises a plurality of elements representing categories of a query~~ of all textual elements corresponding to their order in said mark-up language data stream, said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements and being produced by said streaming API for a mark-up language data stream;

a processor ~~operatively connected to said index and adapted to scan said document~~ adapted to scan all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers; and

~~a parser that is external to said index and said processor and adapted to selectively skip portions of said document based on instructions from said index; and~~

~~a buffer that is operatively connected to said processor and adapted to save said textual categories, wherein said buffer is external to said index, said parser, and said processor~~

a parser adapted to skip an unmatched textual element, if a tag identifier, corresponding to said unmatched textual element, does not match said query.

38. (New) The method of claim 1, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprised a tag identifier and an end position.

39. (New) The method of claim 1, all the limitations of which are incorporated herein by reference, further comprising storing a parsed matched textual element in a buffer.

40. (New) The method of claim 1, all the limitations of which are incorporated herein by reference, wherein upon said skipping of said unmatched textual element for parsing, said method further comprises offsetting said parser based upon an end position stored in said ordered index and corresponding to said unmatched textual element.

41. (New) The method of claim 1, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) or Extensible Markup Language (XML).

42. (New) The system of claim 13, all the limitations of which are incorporated herein by

reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprising a tag identifier and an end position.

43. (New) The system of claim 13, all the limitations of which are incorporated herein by reference, further comprising a buffer, operatively connected to said parser, in which a parsed matched textual element is stored.

44. (New) The system of claim 13, all the limitations of which are incorporated herein by reference, wherein said parser is offset based upon an end position stored in said ordered index that corresponds to said unmatched textual element.

45. (New) The system of claim 13, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) or Extensible Markup Language (XML).

46. (New) The program storage device of claim 25, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprises a tag identifier and an end position.

47. (New) The program storage device of claim 25, all the limitations of which are incorporated herein by reference, further comprising storing a parsed matched textual element in a buffer.

48. (New) The program storage device of claim 25, all the limitations of which are incorporated herein by reference, wherein upon said skipping if said unmatched textual element for parsing, said method further comprises offsetting said parser based upon an end position stored in said ordered index and corresponding to said unmatched textual element.

49. (New) The program storage device of claim 25, all the limitations of which are

incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) or Extensible Markup Language (XML).

50. (New) The system of claim 37, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprising a tag identifier and an end position.

51. (New) The system of claim 37, all the limitations of which are incorporated herein by reference, further comprising a buffer, operatively connected to said parser, in which a parsed matched textual element is stored.

52. (New) The system of claim 37, all the limitations of which are incorporated herein by reference, wherein said parser is offset based upon an end position stored in said ordered index that corresponds to said unmatched textual element.

53. (New) The system of claim 37, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) or Extensible Markup Language (XML).